UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/811,044	03/27/2004	John J. Williams JR.	42027	9536

26327 7590 02/03/2009

THE LAW OFFICE OF KIRK D. WILLIAMS
PO BOX 39425
DENVER, CO 80239-0425

EXAMINER	
HICKS, MICHAEL J	

ART UNIT	PAPER NUMBER
2165	

MAIL DATE	DELIVERY MODE
02/03/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/811,044	WILLIAMS ET AL.	
	Examiner	Art Unit	
	Michael J. Hicks	2165	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *08 November 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-26 Pending.

Response to Arguments

2. Applicant's arguments filed 11/08/2008 have been fully considered but they are not persuasive.

As per Applicants arguments regarding the rejections under USC 101, Examiner respectfully disagrees. Firstly, Examiner notes that the amendments made to Claim 1 overcome the USC 101 rejection applied to Claims 1-5, however Claims 10-11 and 22-26 will continue to stand rejected under USC 101. While Examiner agrees with Applicant in that "…the disclosure of the instant case states that the phrase 'means for xxx' typically includes computer-readable medium or media containing computer-executable instructions for performing xxx," (emphasis added) which is accurate as it includes the word "typically" as there are other embodiments, and one embodiment includes a processor operating according to such computer-executable instructions…" Examiner notes that Applicant admits, in this statement, that the claim may reasonable be construed as "media containing computer-executable instructions for performing xxx." As noted in the rejection of Claims 10-11 and 22-26 under USC 101 found in the Office Action dated 7/9/2008 "…Merely claiming nonfunctional descriptive material, i.e., abstract ideas, stored on a computer-readable medium, in a computer, or on an electromagnetic carrier signal, does not make it statutory. See *Diehr*, 450 U.S. at 185-

86, 209 USPQ at 8 (noting that the claims for an algorithm in *Benson* were unpatentable as abstract ideas because "[t]he sole practical application of the algorithm was in connection with the programming of a general purpose computer.")…". As Claims 10-11 and 22-26 may reasonable be construed as media containing instruction for performing a method, the claims may still be considered to be non-statutory subject matter in light of the above disclosure.

As per Applicants arguments asserting that the lock manager does not forward the protected data and that the data prefetching occurs after acquiring a lock, Examiner respectfully disagrees. Examiner notes that the prefetch operation that Applicants refers to in Optimization 1 of Trancoso related to the initial fetch of data for the initial acquisition of a lock. This is evidenced by the disclosure of Optimization 2 of Trancoso which clearly deals with the situation of forwarding a locked data from a process which currently owns the lock to a requesting process. Examiner further notes that the system presented by Applicant in Applicants claims and specification is expressly described as disallowing the lock manager from accessing the protected data from native storage (e.g. prefetching information). Optimization 1 clearly indicates that the processes prefetch protected data for an initial lock request (note that the claims do not indicate any restriction on the timing of the initial prefetch of data by the first requesting process), and Optimization 2 clearly indicates that the protected data is forwarded from a controlling process to a requesting process. Finally, Examiner notes that the lock manager may also be used to forward protected data as noted in the previous Office

Action as pertaining to Claim 1, such that the releasing process forwards the protected data through the lock manager to the requesting process when communicating with the lock manager to determine the next process in the request queue.

As per Applicants arguments that the data is forwarded from one process to another before the release of the lock, Examiner respectfully disagrees. Examiner notes that Trancoso clearly indicates, as emphasized by Applicant, that the data is forwarded ***right before*** the first processor releases the lock. Examiner interprets this to indicate that the data forwarding instruction is followed adjacently by the release operation. In this interpretation, the data forwarding operation and the lock release operation may, together, be considered to be the totality of the lock release message which is required to contain the protected data.

In light of the above arguments the rejection will be maintained, the standing rejections, with the exception of the rejection of Claims 1-5 Under USC 101, will be maintained.

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. Claims 10-11, and 22-26 rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

As per Claims 10-11, and 22-26, the claims lack the necessary physical articles or objects to constitute a machine or a manufacture within the meaning of 35 USC 101. They are clearly not a series of steps or acts to be a process nor are they a combination of chemical compounds to be a composition of matter. As such, they fail to fall within a statutory category. They are, at best, functional descriptive material *per se*.

Descriptive material can be characterized as either "functional descriptive material" or "nonfunctional descriptive material." Both types of "descriptive material" are nonstatutory when claimed as descriptive material *per se*, 33 F.3d at 1360, 31 USPQ2d at 1759. When functional descriptive material is recorded on some computer-readable medium, it becomes structurally and functionally interrelated to the medium and will be statutory in most cases since use of technology permits the function of the descriptive material to be realized. Compare *In re Lowry*, 32 F.3d 1579, 1583-84, 32 USPQ2d 1031, 1035 (Fed. Cir. 1994)

Merely claiming nonfunctional descriptive material, i.e., abstract ideas, stored on a computer-readable medium, in a computer, or on an electromagnetic carrier signal, does not make it statutory. See *Diehr*, 450 U.S. at 185-86, 209 USPQ at 8 (noting that the claims for an algorithm in *Benson* were unpatentable as abstract ideas because "[t]he sole practical application of the algorithm was in connection with the programming of a general purpose computer."). Note that the drawings pertaining to the rejected claims do not include hardware and thus may be construed as purely software.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1-26 rejected under 35 U.S.C. 102(b) as being anticipated by Trancoso.

As per Claim 1, Trancoso discloses an apparatus for protecting data using locks, the apparatus comprising: a lock manager configured to control access via a lock to protected data maintained in native storage independent of the lock manager (i.e. *"The first optimization attempts to reduce to one the number of accesses to main memory seen by the processor in the critical section. This is done by prefetching right after the acquire all the shared variables that are read in the critical section…There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that a program maintains a queue of processors waiting for a lock, therefore acting as a lock manager. Note that the fact that the lock manager may be used to forward lock variables indicates that the processors may pass data too the lock manager. Also note that the prefetching process is done by the requesting processor, and therefore, the lock manager does not access the protected data in native storage.) (Page III-80, Section 2.1), wherein the lock manager does not access said protected data from said native storage (i.e. *"The first optimization attempts to reduce to one the number of accesses to main memory*

seen by the processor in the critical section. This is done by prefetching right after the acquire all the shared variables that are read in the critical section…There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too." The preceding text excerpt clearly indicates that the prefetching process is done by the requesting processor, and therefore, the lock manager does not access the protected data in native storage.) (Page III-80, Section 2.1); and a plurality, of requesters (i.e. "*There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that a plurality of requestors exist.) (Page III-80, Section 2.1); wherein the lock manager is configured to receive lock requests for the lock from each of the plurality of requesters, and to selectively grant said lock requests which includes communicating grants from the lock manager to the plurality of requesters (i.e. *"The second optimization attempts to hide the single memory access seen by the processor in the first optimization. Indeed, the data to be communicated is forwarded from the first processor to the second right before the first processor releases the lock. These forward statements cost a few cycles to issue. In addition, the issuing processor has to receive an acknowledgement making the completion of each forward before the processor can issue the release."* The preceding text excerpt clearly indicates that grant and release messages are sent to the requesting processors and by the requesting processors, respectively, during the granting and release of locks. Examiner notes that the communications may be routed through the lock manager, as disclosed above,)

(Page III-80, Section 2.1), wherein at least one of said communicated grants includes said protected data (i.e. *"The second optimization attempts to hide the single memory access seen by the processor in the first optimization. Indeed, the data to be communicated is forwarded from the first processor to the second right before the first processor releases the lock. These forward statements cost a few cycles to issue. In addition, the issuing processor has to receive an acknowledgement making the completion of each forward before the processor can issue the release."* The preceding text excerpt clearly indicates that grant and release messages may be accompanied by the requested protected data.) (Page III-80, Section 2.1).

As per Claim 2, Trancoso discloses at least one of said communicated grants does not include said protected data (i.e. *"The first optimization attempts to reduce to one the number of accesses to main memory seen by the processor in the critical section. This is done by prefetching right after the acquire all the shared variables that are read in the critical section…There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that the prefetching process is done by the requesting processor, and therefore, the lock manager does not access the protected data in native storage. As such, the first lock grant message will not be accompanied by protected data, as the requesting processor will be required to prefetch the data.) (Page III-80, Section 2.1).

As per Claim 3, Trancoso discloses said communicated grants include an indication of whether or not said protected data is being communicated therewith (i.e.

"Whenever a lock is acquired, prefetch in exclusive mode all the shared data that will be written within the critical section, and prefetch all the remaining shared data that will be read within the critical section, Right before executing the release operation, if there is a processor waiting to acquire the lock, forward to the processor in exclusive mode all the written shared data and forward to the processor the rest of the rest shared data." The preceding text excerpt clearly indicates that as a requesting processor which is not getting data forwarded to it from a releasing processor must prefetch it's own data, an indication of whether protected data is being transmitted with a grant must be present in order to define the behavior of the requesting processor.) (Pages III-80-III-81, Section 2.1).

As per Claim 4, Trancoso discloses each of said communicated grants includes an indication of whether or not said protected data is requested to be sent to the lock manager with a corresponding release of the lock (i.e. "*There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that the indication of whether or not the protected data is requested to be sent back to lock manager with the lock release is maintained in the list of requestors (e.g. if the list of requestors is empty, the data need not be sent back to the lock manager.).) (Page III-80, Section 2.1).

As per Claim 5, Trancoso discloses each of said lock requests includes an indication of whether or not the corresponding one of the plurality of requesters will accept said protected data from the lock manager (i.e. *"In addition, the issuing processor has to receive an acknowledgement making the completion of each forward before the processor can issue the*

release." The preceding text excerpt clearly indicates that the lock request, at the time of lock issuance, must indicate whether or not the processor will accept the protected data.) (Page III-80, Section 2.1).

As per Claims 6, 8, and 10, Trancoso discloses a method, computer readable medium storing thereon computer executable instructions, and lock manager, for controlling access to protected data maintained in native storage independent of the lock manager (i.e. *"The first optimization attempts to reduce to one the number of accesses to main memory seen by the processor in the critical section. This is done by prefetching right after the acquire all the shared variables that are read in the critical section…There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that a program maintains a queue of processors waiting for a lock, therefore acting as a lock manager. Note that the fact that the lock manager may be used to forward lock variables indicates that the processors may pass data too the lock manager. Also note that the prefetching process is done by the requesting processor, and therefore, the lock manager does not access the protected data in native storage.) (Page III-80, Section 2.1), wherein the lock manager does not access said protected data from said native storage (i.e. *"The first optimization attempts to reduce to one the number of accesses to main memory seen by the processor in the critical section. This is done by prefetching right after the acquire all the shared variables that are read in the critical section…There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this*

queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too." The preceding text excerpt clearly indicates that the prefetching process is done by the requesting processor, and therefore, the lock manager does not access the protected data in native storage.) (Page III-80, Section 2.1), the method comprising: receiving a release of a lock for use in controlling access to said protected data, the release including said protected data (i.e. *"The second optimization attempts to hide the single memory access seen by the processor in the first optimization. Indeed, the data to be communicated is forwarded from the first processor to the second right before the first processor releases the lock. These forward statements cost a few cycles to issue. In addition, the issuing processor has to receive an acknowledgement making the completion of each forward before the processor can issue the release."* The preceding text excerpt clearly indicates that grant and release messages may be accompanied by the requested protected data.) (Page III-80, Section 2.1); identifying a next requester to be granted the lock in response to said receiving the release of the lock (i.e. "*There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that next requestors are identified from the maintained list.) (Page III-80, Section 2.1); copying said protected data from the release into a grant message (i.e. *"The second optimization attempts to hide the single memory access seen by the processor in the first optimization. Indeed, the data to be communicated is forwarded from the first processor to the second right before the first processor releases the lock. These forward statements cost a few cycles to issue. In addition, the issuing processor has to receive an acknowledgement making the completion of each forward before the processor can issue the release."* The preceding text excerpt clearly indicates that grant and release messages are sent to the

requesting processors and by the requesting processors, respectively, during the granting and release of locks, and may include the protected data in the form of data forwarded during the grant and/or release procedure. Examiner notes that the communications may be routed through the lock manager, as disclosed above,); and sending the grant message to the next requester, the grant message including said protected data (i.e. *"The second optimization attempts to hide the single memory access seen by the processor in the first optimization. Indeed, the data to be communicated is forwarded from the first processor to the second right before the first processor releases the lock. These forward statements cost a few cycles to issue. In addition, the issuing processor has to receive an acknowledgement making the completion of each forward before the processor can issue the release."* The preceding text excerpt clearly indicates that grant and release messages are sent to the requesting processors and by the requesting processors, respectively, during the granting and release of locks, and may include the protected data in the form of data forwarded during the grant and/or release procedure. Examiner notes that the communications may be routed through the lock manager, as disclosed above,).

As per Claims 7, 9, and 11, Trancoso discloses the grant message includes an indication of that said protected data is requested to be sent to the lock manager in a release message corresponding to the grant message if another requester is waiting for the lock (i.e. "*There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that the indication of whether or not the protected data is requested to be sent back to lock manager with the lock release is maintained in the list of requestors (e.g. if the list of requestors is empty, the data need not be sent back

to the lock manager.).) (Page III-80, Section 2.1), else an indication that said protected data is not requested to be sent to the lock manager in the release message (i.e. *"To handle the case where no processor is waiting, we have to combine this optimization with the Pref optimization discussed above: before issuing the forwards, the processor has to check if there is any processor waiting. If none is waiting, the forwards are skipped"* The preceding text excerpt clearly indicates that the indication of whether or not the protected data is requested to be sent back to lock manager with the lock release is maintained in the list of requestors (e.g. if the list of requestors is empty, the data need not be sent back to the lock manager.).) (Page III-80, Section 2.1).

As per Claims 12, 17, and 22, Trancoso discloses a method, computer readable medium storing thereon computer executable instructions, and lock manager performed by a lock manager controlling access to protected data maintained in native storage independent of the lock manager (i.e. *"The first optimization attempts to reduce to one the number of accesses to main memory seen by the processor in the critical section. This is done by prefetching right after the acquire all the shared variables that are read in the critical section…There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that a program maintains a queue of processors waiting for a lock, therefore acting as a lock manager. Note that the fact that the lock manager may be used to forward lock variables indicates that the processors may pass data too the lock manager. Also note that the prefetching process is done by the requesting processor, and therefore, the lock manager does not access the protected data in native storage.) (Page III-80, Section 2.1), wherein the lock manager

does not access said protected data from said native storage (i.e. *"The first optimization attempts to reduce to one the number of accesses to main memory seen by the processor in the critical section. This is done by prefetching right after the acquire all the shared variables that are read in the critical section…There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that the prefetching process is done by the requesting processor, and therefore, the lock manager does not access the protected data in native storage.) (Page III-80, Section 2.1), the method comprising: receiving locking requests for a lock controlling access to said protected data from a first requester and a second requester (i.e. "*There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that a plurality of requestors exist.) (Page III-80, Section 2.1); sending a first grant message to the first requester, the first grant message not including said protected data (i.e. *"The first optimization attempts to reduce to one the number of accesses to main memory seen by the processor in the critical section. This is done by prefetching right after the acquire all the shared variables that are read in the critical section…There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading*

such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too." The preceding text excerpt clearly indicates that the prefetching process is done by the requesting processor, and therefore, the lock manager does not access the protected data in native storage. As such, the first lock grant message will not be accompanied by protected data, as the requesting processor will be required to prefetch the data.) (Page III-80, Section 2.1), and in response to identifying one or more requesters is waiting for the lock after the first requester, including an indication to return said protected data in the first grant message (i.e. "*There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that the indication of whether or not the protected data is requested to be sent back to lock manager with the lock release is maintained in the list of requestors (e.g. if the list of requestors is empty, the data need not be sent back tot he lock manager.).) (Page III-80, Section 2.1); and receiving a first release message corresponding to the first grant message for the lock from the first requester, the first release message including said protected data (i.e. *"The second optimization attempts to hide the single memory access seen by the processor in the first optimization. Indeed, the data to be communicated is forwarded from the first processor to the second right before the first processor releases the lock. These forward statements cost a few cycles to issue. In addition, the issuing processor has to receive an acknowledgement making the completion of each forward before the processor can issue the release."* The preceding text excerpt clearly indicates that grant and release messages may be accompanied by the requested protected data.) (Page III-80, Section 2.1).

As per Claim 13, 18, and 23, Trancoso discloses sending a second grant message to the second requester, the second grant message including said protected data (i.e. *"The second optimization attempts to hide the single memory access seen by the processor in the first optimization. Indeed, the data to be communicated is forwarded from the first processor to the second right before the first processor releases the lock. These forward statements cost a few cycles to issue. In addition, the issuing processor has to receive an acknowledgement making the completion of each forward before the processor can issue the release."* The preceding text excerpt clearly indicates that grant and release messages may be accompanied by the requested protected data.) (Page III-80, Section 2.1), and an indication of whether or not to send said protected data in a second release message (i.e. "*There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too."* The preceding text excerpt clearly indicates that the indication of whether or not the protected data is requested to be sent back to lock manager with the lock release is maintained in the list of requestors (e.g. if the list of requestors is empty, the data need not be sent back to the lock manager.).) (Page III-80, Section 2.1).

As per Claim 14, 19, and 24, Trancoso discloses the second grant message includes an indication to send said protected data in the second release message in response to identifying another requestor is waiting for access to the lock (i.e. "*There are some problems with this use of data forwarding. The first one is that the processor issuing the forwards needs to know which other processor to send the data to. The most obvious way of doing this is to keep in memory a queue of all the processors waiting for that lock. Then, the releasing processor can read this*

queue to determine what, processor to forward the data to. Reading such queue, of course, involves some overhead. We note in passing that this queue can be used to forward the lock variable itself too." The preceding text excerpt clearly indicates that the indication of whether or not the protected data is requested to be sent back to lock manager with the lock release is maintained in the list of requestors (e.g. if the list of requestors is empty, the data need not be sent back to the lock manager.).) (Page III-80, Section 2.1).

As per Claim 15, 20, and 25, Trancoso discloses the second grant message includes an indication not to send said protected data in the second release message in response to identifying another requestor is not waiting for access to the lock (i.e. *"To handle the case where no processor is waiting, we have to combine this optimization with the Pref optimization discussed above: before issuing the forwards, the processor has to check if there is any processor waiting. If none is waiting, the forwards are skipped"* The preceding text excerpt clearly indicates that the indication of whether or not the protected data is requested to be sent back to lock manager with the lock release is maintained in the list of requestors (e.g. if the list of requestors is empty, the data need not be sent back to the lock manager.).) (Page III-80, Section 2.1).

As per Claim 16, 21, and 26, Trancoso discloses wherein the second grant message includes an indication not to send said protected data in the second release message (i.e. *"To handle the case where no processor is waiting, we have to combine this optimization with the Pref optimization discussed above: before issuing the forwards, the processor has to check if there is any processor waiting. If none is waiting, the forwards are skipped"* The preceding text excerpt clearly indicates that the indication of whether or not the protected data is requested to be sent back to lock manager with the lock release is maintained in the list of requestors (e.g. if the list of requestors is empty, the data need not be sent back to the lock manager.).) (Page III-80, Section 2.1).; and the

method comprises in response to said indication not to send said protected data in the second release message, the second requester storing said protected data and not including said protected data in the second release message (i.e. *"To handle the case where no processor is waiting, we have to combine this optimization with the Pref optimization discussed above: before issuing the forwards, the processor has to check if there is any processor waiting. If none is waiting, the forwards are skipped"* The preceding text excerpt clearly indicates that the indication of whether or not the protected data is requested to be sent back to lock manager with the lock release is maintained in the list of requestors (e.g. if the list of requestors is empty, the data need not be sent back to the lock manager.). Examiner notes that if the data is not forwarded, it will be stored, as no other processes require the data.) (Page III-80, Section 2.1).

7. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Points of Contact

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Hicks whose telephone number is (571) 272-2670. The examiner can normally be reached on Monday - Thursday 9:00a - 7:30p.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Christian Chace can be reached on (571) 272-4190. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Michael J Hicks
Art Unit 2165
Phone: (571) 272-2670
Fax: (571) 273-2670

/Christian P. Chace/

Supervisory Patent Examiner, Art Unit 2165